3D HUMAN POSE ESTIMATION

A Dissertation
Presented to
The Academic Faculty

By

Ashar Ali

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2019

3D HUMAN POSE ESTIMATION

Approved by:

Dr. Patricio A. Vela, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Ayanna M. Howard
School of Interactive Computing
College of Computing
Georgia Institute of Technology

Dr. Anthony J. Yezzi
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Irfan Essa
School of Interactive Computing
College of Computing
Georgia Institute of Technology

Date Approved: April 26, 2019

Practice puts brains in your muscles.

Sam Snead

Dedicated to Shikwat Ali, Asiya Chaudhary and Sariya Ali for their thorough support and motivation.

ACKNOWLEDGEMENTS

I would like to sincerely express my gratitude to my advisor Dr. Patricio Vela, for his persistent and valuable feedback. His guidance has always made me ask the right questions about each aspect of my research, and steered me into the right direction whenever required.

I would like to make a special mention about my mentor and friend Miguel for introducing me to the field of 3D Pose Estimation, and constantly pushing my limits throughout the initial phase when I built my research acumen.

I would also thank all the committee members Prof. Ayanna Howard, Prof. Irfan Essa and Prof. Anthony Yezzi for their precious time.

Also, I would like to mention my lab mates Fujen, Justin, Abhishek and all the others. It has been a fun ride working alongside these people who are always ready for discussing and brainstorming about new ideas.

In the end, I owe my deepest gratitude to my family's unconditional support. My parents, Shikwat and Asiya, and my sister Sariya have always backed me in times of extreme stress and joy alike.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of the proposed work is to understand how using synthetic datasets and automatic annotation policies can further state of the art research for 3D Human Pose Estimation. Given an input depth image, algorithms estimating 3D human pose can be grouped into two major categories. Some directly regress 3D joints from depth images (or corresponding point clouds / voxels), often referred to as Single-Stage approaches. Others resort to Two-Stage solutions, where they first segment these depth images with dense labels and then use corresponding (segmented) point clouds to regress joint coordinates in 3D. Contribution of work proposed in this thesis would be three-fold. First, we demonstrate that present Two-Stage approaches can be improved using automated labelling techniques on real as well as synthetic datasets individually. Second, a novel Single-Stage algorithm is designed, which inputs human point cloud and outputs the pose in 3D world coordinates. Finally, this work studies the impact of fusing synthetic datasets with real datasets while training.

CHAPTER 1
INTRODUCTION AND BACKGROUND

Modern day computer vision relies heavily on deep learning techniques. From fundamental tasks like object recognition [1], [2], detection [3], [4] and tracking [5], [6] to high level semantic dilemmas like traffic scene understanding [7], [8] , the community has witnessed substantial performance boost with these algorithms. Latest GPU hardware provide immense computational capabilities, which enable rapid prototyping and deployment of new ideas. Deep Learning methods have proven themselves to generalize much better than many hand-engineered efforts, but only as good as generalizability of the training data itself. Data annotation is increasingly becoming the most expensive part of developing such algorithms, in terms of cost as well as in terms of time requirements.

This shortage in availability of real datasets is being addressed in multiple ways. The most prominent among these are advancements in the computer graphic simulators, which have become extremely good at generating realistic images, pointclouds, CAD objects, etc. Open source software like Blender, NVIDIA Deep Learning Dataset Synthesizer (NDDS) [9] have been widely used for generating 2D/3D images for objects with configurable meta information like camera parameters, colors, object pose and textures. Apart from the synthetic data generation software, groundbreaking research in the form of Generative Adversarial Networks [10] has also enabled synthesis of 'realistic' fake data after learning from inherent representations in available real datasets. Just as synthetic data generation for additional data points is being actively developed, another field of work which has been fueled recently by deep learning techniques is Domain Adaptation, which has empowered algorithms to ensure better transferability of information from clean training datasets to complex real world setups.

Human Pose Estimation draws a considerable attention in computer vision community

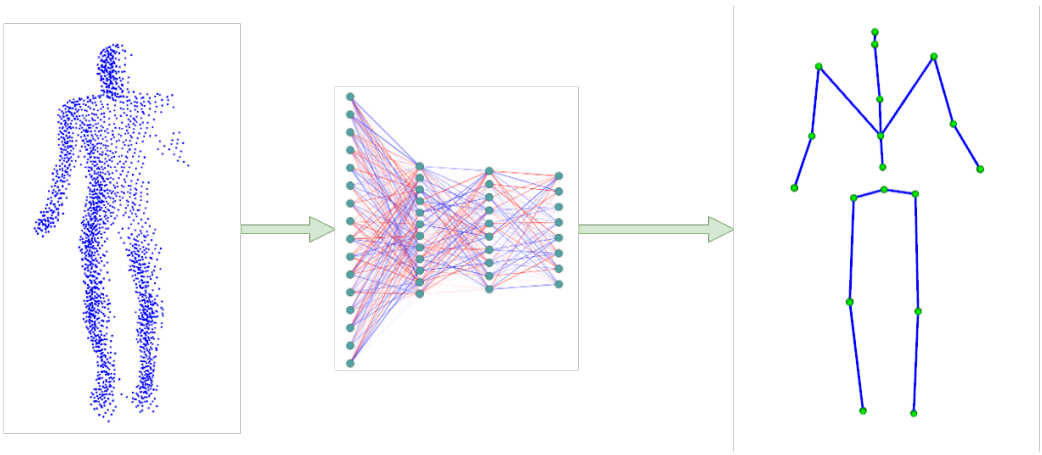


Figure 1.1: Proposed high-level architecture for 3D Human Pose Estimation

because of an array of applications in augmented reality, activity recognition, trajectory prediction, and marker-less tele-operation to name a few. Depending upon the application, requirement for pose estimation can be modelled into three categories- 2D Pose Estimation (in pixel coordinates) from static monocular images, 3D Pose Estimation (in world/camera coordinates) from depth/range images, and 3D Pose Estimation directly from monocular/2D images. Depth sensors have the ability to provide rich information about human postures in indoor settings, and are used in multiple setups, like gaming in Microsoft's Xbox with Kinect, etc. But flexibility of the human body as well as multiple degrees of freedom leading to self-occlusion has kept pose estimation of humans far from solved.

Under the purview of problems faced by the bigger computer vision community, 3D Human Pose Estimation also suffers from the lack of sufficient datasets for training with real human subjects. This is because it requires a big controlled setup with motion capture systems like Optitrack, which are expensive to purchase in the first place. This work entails controlled experiments to study how semi-automated annotation and the use of synthetic datasets can increase the performance of existing techniques. Also, this work contributes to the existing literature of 3D Human Pose Estimation with a novel single-stage algorithm which performs better in terms of efficiency, and is free from a lot of hassles involved in present techniques like intermediate 3D reconstruction. Fig.1.1 demonstrates this architec-

ture, which takes in a sub-sampled pointcloud directly from a depth sensor, and estimates the 3D posture as an output.

This thesis has been structured in the following chapters to discuss in detail about literature review, methods and technologies developed, discussion about experiments performed on different datasets, and conclusion with an analysis of results. The methodology section in Chapter 3 of the thesis unfolds in three stages. First, an automated annotation policy is discussed to assign intermediate dense-labels for semantic segmentation of depth data into multiple body parts. It involves the use of nearest-neighbor approach based on the mocap skeleton groundtruth. This automatic supervision shows that even real datasets can be richly annotated for strong intermediate feature extraction in the latest pose estimation techniques. Second, a novel single-stage algorithm is described and shown how it best addresses the space-time efficiency vs accuracy trade-off in 3D Human Pose Estimation on depth maps from single as well as multiple views. Finally, a detailed and controlled study is performed to analyze the impact of augmenting/replacing real data with compatible synthetic datasets on overall performance and accuracy of this single-stage algorithm.

CHAPTER 2
LITERATURE REVIEW

In this literature review, we focus on how present techniques tackle human pose estimation in both 2D and 3D. We identify key technological improvements in 2D and 3D Computer Vision which motivate us to build our own algorithm. We also identify the key areas which are closest to practical applications in real world, and how limited variability in present real datasets can be addressed using their synthetic counterparts. Broadly, human pose estimation is categorized into three problem domains.

2.1 Human Pose Estimation Techniques

2.1.1 2D Human Pose Estimation (with monocular input)

Numerous techniques have been developed to find pixel coordinates ***(xi , yi)***, corresponding to each joint ***i*** in static images. Broadly, they are classified as generative, discriminative or hybrid.

Generative Approaches

Remarkable first attempts at 2D pose estimation in video domain date back to 1980 when O Rourke and Badler [11] proposed a top-down approach, assuming human body as a tree-like connected model. Another approach by Fischler and Elschlager [12], called the pictorial structures proposed a deformable parts model in which the appearance of each part is modeled individually and pairwise relationships are modeled by spring-like connections between pairs of parts. This work provided for development of general frameworks for structural object detection [13], [14], as well as influential breakthroughs in the field of 2D Human Pose Estimation [15], [16]. As a whole, all these generative approaches involve

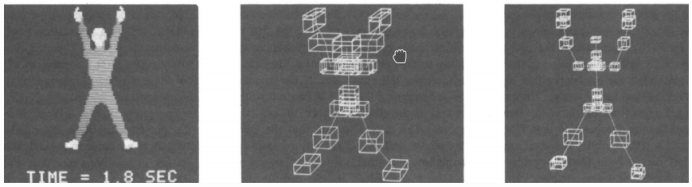


Figure 2.1: O Rourke and Badler's Tree-Based Model

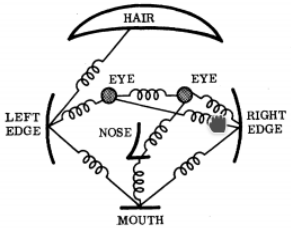


Figure 2.2: Pictorial Structure Approach for detecting face keypoints

encoding a parametric model which serves as a prior to learn spatial relationships between different body parts, and optimized graphical inference techniques are employed for final pose estimation.

Discriminative and Hybrid Approaches

Bottom-up part-based models like articulated structures by Ramanan [15], derive inspirations from iterative parsing processes on CRFs. Another hybrid approach by Ramanan [16] where human body is modelled as a flexible mixture of parts was one of the very influential works in 2D Human Pose Estimation. However, with recent breakthroughs in deep learning techniques since 2012, powerful discriminative Convolutional Neural Network (CNN) models proposed by Szegedey [17], stacked hourglass networks by Newell [18] and Openpose by Cao [19], along with [20], have outperformed all the previous approaches based on hand- engineered feature extraction. Most of these work on the principle of regressing heatmaps which are a Gaussian distribution of specific joint locations in pixel space. Convolutional Pose Machines [21], and Stacked Hourglass algorithms introduced the use of multiple stacked networks with intermediate losses in order to deal with the problem of vanishing gradients for the first time. This enabled the networks to learn complex situations

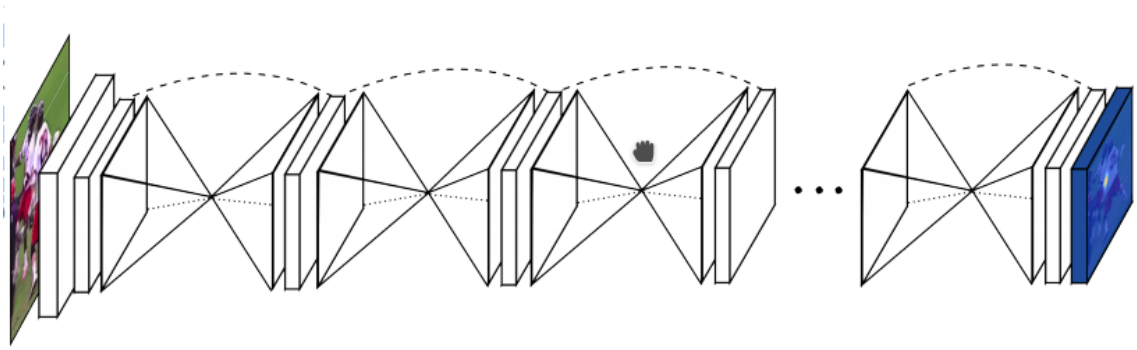


Figure 2.3: Stacked Networks learning Pose heatmaps as an intermediate step to estimate 2D poses

incorporating multiple cases of self as well as environmental occlusion.

2.1.2 3D Human Pose Estimation (with monocular input)

Many techniques like [22], [23], [22], [24], [25] have attempted to obtain 3D pose estimates directly from monocular input, after the Humans 3.6M dataset [26] was published. Two-stage approaches like [22], [23] estimate 2D (pixel-space) poses from monocular images using 2D pose estimation as a first step. In the second step, 3D joints in a camera relative frame are lifted from 2D pixels, either by a constrained deep regression [22], [27] or by matching 2D poses with 2D projections obtained from a library of existing 3D poses. Other papers directly regress 3D joint locations after extracting visual features from images [28], [24], [29]. A group led by Pascal Fua presented [30] where a learnable fusion of network parameters predicting 2D joint locations and extracting 3D cues from the image is executed, towards the final goal of predicting a vector representing 3D coordinates of the skeleton. Recently, a multitask deep-learning framework for pose-estimation and activity recognition [25] introduced a volumetric heatmap criterion. The volumetric (3D) heatmaps are then further processed with a Soft-argmax function to estimate final 3D pose. A very exciting work presented in [31] solves this problem in a geometry aware fashion, wherein the process is broken down into two steps. First, an autoencoder learns a latent 3D representation of the human. Then, this reconstructed 3D human is used to learn 3D poses in a

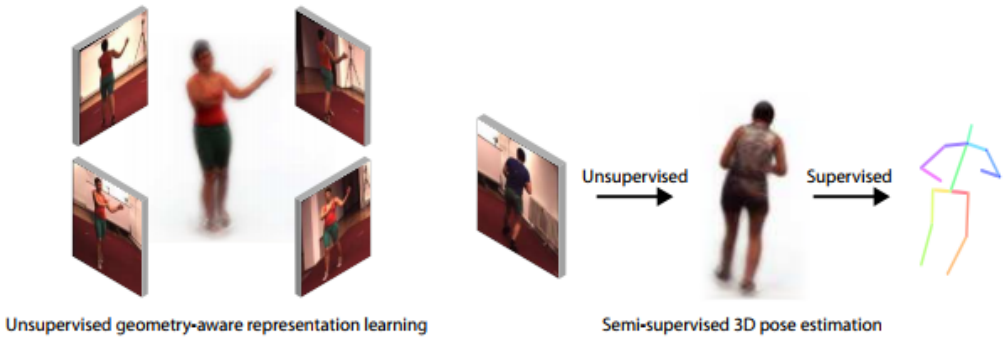


Figure 2.4: Geometry aware semi-supervised method for 3D Human Pose Estimation

supervised setup. Unfortunately, all approaches which estimate 3D joint coordinates from images are in a root-relative (relative to torso) coordinate frame and cannot be brought to use in real world 3D scenarios. Therefore, these are all learning the intrinsic parameters of the camera and are far from use in real applications.

2.1.3 3D Human Pose Estimation (with depth input)

Low-cost depth sensors have had an enormous impact on consumer markets (like Microsoft XBOX) as well as multiple robotics applications like on drones for indoor/outdoor navigation, etc. Just like their monocular counterpart, depth-based 3D Pose estimation techniques can also be classified as either Two-Stage or Single-Stage. Shotton and Girshick [32] used Random Forests to classify each pixel into body parts, and subsequently derived joint locations from these maps with a Mean-Shift based approach. Other two-stage approaches like [33] use depth map labels combined with probabilistic graphical models for final inference involving graph cuts. Modern deep-learning approaches like [34] first segment depth maps using CNNs, and then reconstruct segmented point-clouds to obtain 3D joint locations. A library of 3D poses is also used in some approaches for fitting into the outputs of a segmented depth map, instead of directly regressing, as in [35]. Although this approach claims to observe exceptionally good results, but still fails in cases where poses are unexpectedly different at test time.

Single-stage approaches like [36] directly derive a point cloud, and then use that point

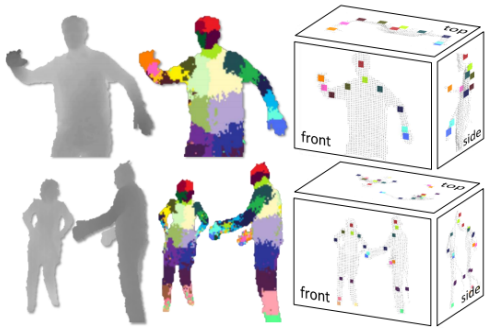


Figure 2.5: Shotton et. al's Pose Estimation Algorithm for Kinect: Involves semantic segmentation of body parts as an intermediate stage.

cloud to match with pre-defined exemplars, which are further refined for location estimates. Modern single-stage approaches like DDP [37] directly regress 3D joints from depth maps using CNNs and then model this inference as a linear combination of bases of 3D poses learned from the training dataset. Other single-stage techniques like V2V Posenet [38] treat derived voxels as their inputs to regress joint locations with 3D CNN-autoencoders. These voxels are slow to generate in the first place, and the resulting inference time for one example is around 3.5 fps for an ensemble of 10 models to obtain the results.

2.2 Relevant Contemporary Advancements in 2D/3D Computer Vision

Very few works in the literature have demonstrated use of semi-automatic annotation policies on limited datasets like [39], [40]. Some other works like [34], [37] use synthetic and real datasets individually to benchmark their algorithms. A lot of studies in recent literature [41], [42], [43] show that using synthetic datasets while training can boost the learning performance. But a controlled study on what impact these techniques could have in improving human pose estimation on real datasets cannot be found in existing literature. Also, a very recent development in 3D computer vision known as PointNets [44] has revolutionized multiple recognition and detection tasks in 3D in a space-efficient and rotation-invariant manner. Therefore, a new single-stage neural network architecture inspired from PointNets is proposed to estimate 3D joint coordinates from depth data. Another key contribution of

this work would be to understand the impacts of combining real and synthetic datasets in improving human pose estimation on real human datasets.

CHAPTER 3
TECHNIQUES IMPLEMENTED AND EVALUATED

This section primarily describes present state of the art techniques which were implemented and evaluated in order to set benchmarks for both - Synthetic UBC (Hard) and Real MHAD Datasets. As described in the literature review section above, 3D Human Pose Estimation can be classified into two type of technologies, Single-Stage and Double-Stage, and the best performing algorithms were picked in both categories to benchmark the performance of proposed automatic annotation policies extensively.

3.1 Datasets Used

3.1.1 *UBC Synthetic Human Dataset*

The UBC Dataset comprises of depth images sub-categorized into either of the easy, medium and hard datasets. The structure of this dataset is inspired from the curriculum learning [45] theory which states the fact that just like humans, it is easier for machines to learn tasks in the order of increasing complexity. The claimed difficulty can be directly correlated with the deviation of body pose from a standing upright position. Experiments in this thesis primarily focus on the Hard section of the UBC set. This dataset was generated with help of Make Human open-source software, and includes a wide variety of complex poses. These poses are actually 100,000 most dissimilar cluster centroids obtained from the publicly available CMU Motion Capture Dataset.

Using the pose as prior, samples of depth images were generated for three different camera viewpoints. In addition to poses, the characters simulated also include variations in clothing, age, gender and shape. Locations of the three different camera are also randomly sampled from an interval of $[-\pi, \pi)$ in azimuthal space. Several instances of camera loca-

Figure 3.1: Depth Images and body part labels provided in the 3-view UBC Dataset.

tion are displayed in Fig. 3.1. For this UBC dataset, number of joints for a complete body posture is 18, as displayed in Fig. 3.2 The ground truth locations of each joint are relative a coordinate frame where torso of the person is always at the origin. The UBC-Hard dataset, also comes with a segregation into three sets- train, test and validation splits. The train set consists of 59 subjects, validation has 20 and test set has a total of 19 persons. For each person in all the three sub-divisions, there are 1001 images captured from 3 different views, as demonstrated above. Thus, the full dataset has 177177 depth images in train, 60060 images in validation and 57057 images in the test set. The resolution of each image is 640 x 480 pixels, with intensities in the range of $[0, 255]$. The ground truth meta-data provides rotation and translation values and an intrinsic camera parameter matrix corresponding to each frame. To visualize the correspondence of ground truth skeletons, these intrinsic-extrinsic parameters are used for point cloud reconstruction, which can aggregate as shown in the Fig. 3.2.

3.1.2 *MHAD Real Human Dataset*

Berkeley Multi-Modal Human Action (MHAD) dataset was collected for real humans. A total of 7 male and 5 female subjects in the age-range of 25-30 years were captured, except for one male subject who was elderly. Each subject was asked to perform a total of 11 actions. For consistency, each action was repeated a total of 5 times. As the name suggests, this dataset has synchronized recordings of its subjects from multiple sensors, including cameras, depth sensors, accelerometers and also microphones. For skeleton tracking

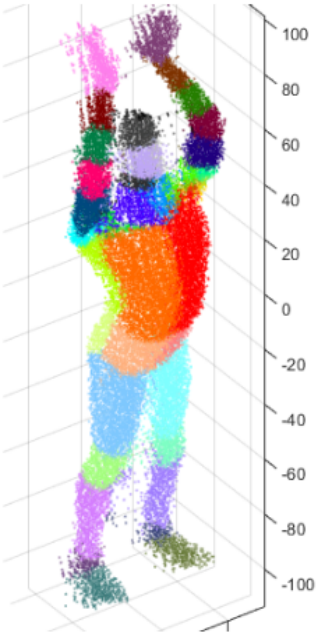


Figure 3.2: Point Clouds reconstructed from region labels and Depth Images

purposes, motion capture system provided by Impulse Systems, which efficiently tracks skeletons at a frequency of 480 GHz.

For relevance of this thesis, two Kinect sensors are used for acquisition of the depth data, one is placed at front of the subject and another is at the back. Data streams recorded from all these sensors are synchronized on the server using an NTP client service. The different actions involve movements of subjects with variations of pose, and two actions also involve a background object (chair), which is used by subjects to sit and stand, provided a lot of clutter in depth feeds. An instance of this dataset can be found in the fig. 3.3. The total number of depth images involving 2 kinect sensors, 12 subjects, 11 actions, and 5 video recordings is more than 250,000.

Due to real captures, this data also has a lot of clutter, which could inherently worsen results for pose estimation problem. Intrinsic and extrinsic camera parameters are first used for visualizing these subjects as pointclouds (an example shown in Fig. 3.3). Then, as a preprocessing step, the ground floor is removed using RANSAC plane fitting. Also, point cloud only in vicinity of the human is extracted, so as to further get rid of all objects in the

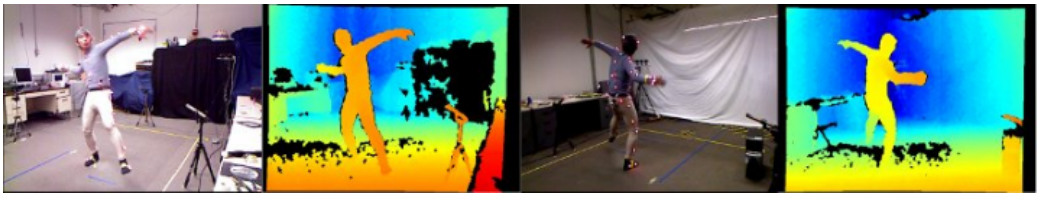


Figure 3.3: Data captured from two vies in the MHAD Dataset

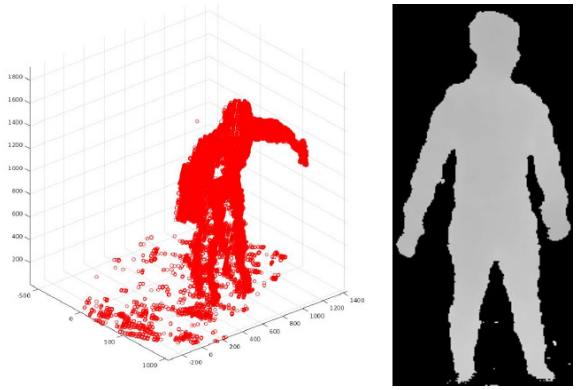


Figure 3.4: 1. PCL after removing ground plane and background clutter 2. Back projected depth image further preprocessed to remove all noise in the frame to get a tight bounding box on the human.

background. An example of a depth image after preprocessing looks like the one shown in Fig. 3.4. This version also incorporates removing all other background and just extracting human within a bounding box to retain cleaner data for improved results.

3.2 Two - Stage Methods

Most interesting two-stage approaches appeared right after Kinect's pose estimation algorithm was published [32]. The algorithm first segments pointcloud into different regions using random forests. And then, these regions are further processed with a mean-shift algorithm, in order to generate refined results for 3D keypoints. A very recent development on the same lines can be found in Shafei et. al [34]. Although this work provides trained models to reproduce its test results, but the training pipeline was not made publicly available. Therefore, a similar framework was implemented as mentioned in their paper in order to benchmark results on their dataset. The whole process can be broken down into two stages,

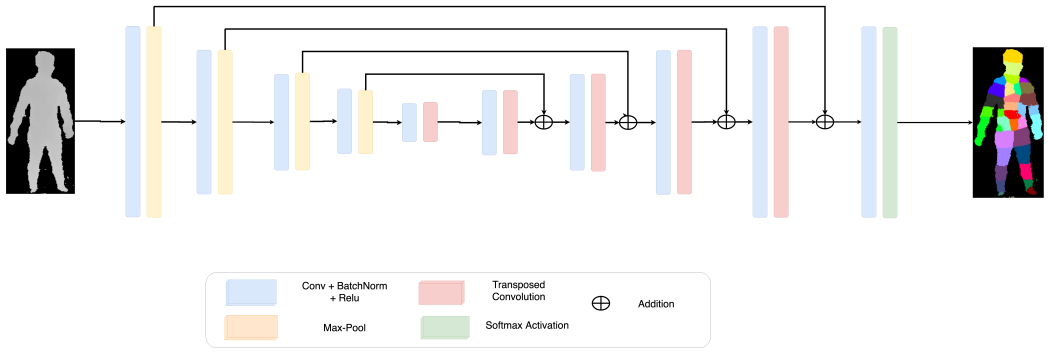


Figure 3.5: Semantic segmentation architecture for the first stage

as described below.

3.2.1 Stage - 1: Region Classification and Point Cloud Reconstruction

Semantic Segmentation

First part of the model takes in depth images of humans, and segments that into multiple body regions. This task is realized with a deep neural network for semantic segmentation. As mentioned in the paper, this network is an adaptation of Fully Convolutional Netwokrs (FCN) [46]. Resembling an autoencoder framework, this network comprises of an encoding layer built off of VGG-16, stitched to a decoding layer which is a lateral inversion of the encoding layer. For the encoder part, only the first 13 layers of VGG-16 are truncated and all layers are convolutions of size 3 x 3, followed by Batch-normalization, ReLU activation and Max-Pooling layers. Encoders can be understood as strong feature extractors at different levels of receptive fields. Decoder part of the network is implemented using the same 13 layers, but in an opposite order. A more technical way to understand their functionality is to declare them as transposed-convolutional layers. Semantically, decoding part of the architecture becomes responsible for object shape generation. Input layer of this network requires an image of resolution 224 x 224 x 3 pixels, and the output softmax layer has a dimension of 224 x 224 x *number of classes*. Each layer in the output provides a probability that a pixel belongs to that class.

A visualization of this network is provided in the Figure 3.5. As can be clearly observed

in the visualization, some body regions are smaller in size as compared to others. Also, number of pixels in the background class is much more than any other part labeled as a body region. This leads to a class imbalance problem, and the network easily pitfalls into the zone where it clearly learns the distinction between background and foreground, but it is unable to discriminate about different body regions, especially those which are smaller in size. In order to handle this imbalance, median frequency balancing was incorporated in the loss function. Individual weights of each class correspond to the ratio of the median of all class frequencies computed on the entire training set divided by the particular class frequency. This implies that larger classes in the training set have a weight smaller than 1 and the weights of the smallest classes are the highest. E.g, for class c, weight associated with class c in the loss function can be defined as

$$\alpha_c = \textit{median of all class frequencies}/\textit{frequency(c)}$$

Although median frequency balancing compelled the network to converge, there was still a discrepancy in learning smaller regions. Significant clutter was observed because the upsampling layers were not strong enough to interpolate distinct boundaries for regions from smallest layers of abstraction in the network. Therefore, skip connections inspired from the ResNet architecture were introduced into this network to retain information about regions smaller than the others. Output from each convolutional layer in the encoding portion of network was added to corresponding layer in the decoding portion. The introduction of these skip connections also increased the gradient flow in the network, and ultimately helped to converge faster. Eventually, the network design described in Fig. 3.5 was used for final training and testing of all the dataset combinations.

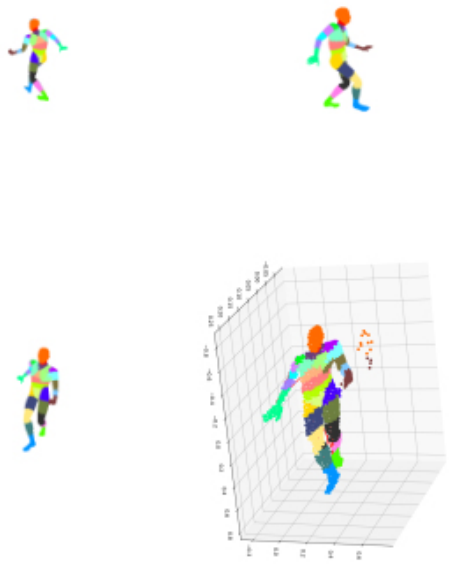


Figure 3.6: Aggregation of three views to form pointclouds

Multiple View Aggregation

At this step, first part of the model has already generated n different outputs which are densely classified. Now the second stage requires a single labeled 3D pointcloud for pose estimation. Using the intrinsic camera parameters, these depth images are first used to reconstruct an individual point cloud for each of the n cameras. Next, these individual pointclouds are merged with the help of provided extrinsic parameters in order to form a unified pointcloud relative to a desired location.

3.2.2 Stage - 2: Feature Extraction and Regression

Using the point cloud information, first and second order moments of each body part location are calculated. For each of the three dimensions, feature vector consists of (i) Median of the values, (ii) Standard Deviation (iii) Minimum and Maximum (iv) Covariance Matrix and (v) Eigenvalues of the co-variance matrix. A simple feature set ensures a high throughput in the feature extraction pipeline. Thus a total of 24 features are extracted for all the

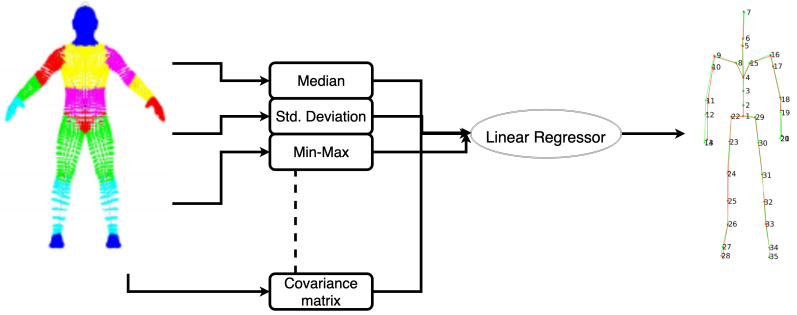


Figure 3.7: Linear Regression pipeline for final pose estimation

joint locations, providing a feature vector $f \in R^n$ where n = 24 x number of segmented body parts. These features are used to learn a set of linear regression models for one co-ordinate value of each joint location. For example, 18 joints of a skeleton would result in a total of 54 regression models. Each of these models can be understood as optimizers under a least-squares formulation. Both L_2 and L_1 regularization were tried, but it had a negligible difference in performance, and L_1 regularizer took substantial extra time. Thus, L_2 was selected as a design choice for this experiment. A demonstration of this regression pipeline can be viewed in Fig. 3.7.

3.2.3 Automated Annotation

The results on two datasets described above were analyzed for this two-stage process. One important point to note here is that the real MHAD dataset does not have any dense region labels associated with its depth images. Authors of the above two-stage pipeline completely leveraged synthetic UBC data to learn their semantic segmentation model. Body region labels are inferred, and then a single labeled point cloud is generated for the real data instances. Thus, second stage of the pipeline remains unchanged for two datasets.

Since the two datasets belong to different modalities, an inherent bias is involved when inferring body regions of real MHAD dataset in the first stage. This clearly indicates that if semantic segmentation model was able to learn on the real data itself, it could perform much better pose estimation on new real data. Based on this intuition, an automated an-

notation policy is proposed. The main motivation is to be able to generate body region labels without any manual intervention. A two-step process is designed to generate these labels automatically. First, camera's intrinsic and extrinsic parameters are used to generate pointcloud in world reference coordinates. Now, distance of each point in the pointcloud is calculated with n skeleton joint locations. Region label corresponding to the nearest joint is associated with each point in 3D. For the second step, these labels in 3D are projected back to the real image dimensions in 2D. This two-step process can also be directly related with the nearest neighbor approach in traditional machine learning.

This automation for intermediate region labeling enables training directly on real data, and provides a mechanism to get rid of the bias inherent in the first stage. Results and discussion in Chapter 5 demonstrate that total error is reduced by up to 10 %. Further discussion about automated labeling follows in chapter 5.

3.3 Single-Stage Methods

The most recent works in single-stage pose estimation approaches [37], [38] either use depth images directly for inference, or first discretize the generated point clouds into voxels and then estimate a pose in 3D world coordinates. Presently, the Deep-Depth Pose algorithm [37] claims to have the best results among two stage approaches on the synthetic dataset UBC3V.

The authors have provided trained models for inference on a dataset called ITOP [47], where they achieve a 100 % *accuracy at 10 cm*. The primary idea stated is given a depth image, it is possible to directly regress 3D joint coordinates of a human in 3D world space. Trained models for inference on UBC 3V were not provided, so an implementation of this approach, as described below was attempted.

3.3.1 Deep Depth Pose Model

Numerous approaches for 3D estimation of poses from 2D images follow a pattern, often termed as library-based matching. They would first infer 2D poses from the image, and with inverse projection, multiple 3D poses are proposed. These 3D candidates are then compared with all the existing 3D poses in library mentioned above. This idea forms an inspiration for our DDP approach. Here instead of a full-fledged library, a small collection of poses is obtained. But this is not selected from a library of actual 3D poses. Instead, all the poses in existing training data are clustered, and these cluster centroids serve as bases for a high-dimensional vector space. These bases are also termed as 'prototypes' in the original paper.

This single stage approach assumes that a pose can be approximated by a linear combination of *K* prototype poses as described above.

$$\mathbf{P} = w_1 * C_1 + w_2 * C_2 + .. + w_k * C_k$$

where w_i is the weight assigned to the prototype C_i. Therefore, this approach learns to regress the weight vector **w**.

The weight vector is learned with an implementation of a Convolutional Neural Network, inspired from the AlexNet [48] architecture. Input to this network is a single depth image of resolution 100 x 100, and each convolutional block is followed by ReLU activation. Some later layers have max-pooling, eventually followed by fully connected layers. The first fully connected layer also uses a dropout for regularization purposes. The exact dimensions of convolutional kernels can be referred in Table 3.1. Input layer expects an image of dimensions 100 x 100, which is obtained after multiple stages of preprocessing. All images are first projected to a pointcloud, and similar to the two-stage method, they are first pre-processed to remove any background clutter. Again, cleaner images are projected back to the image plane, and then min-max normalization brings their intensities down to

Table 3.1: Deep Depth Pose Architecture

Layer	**Dimensions**
Input	100 x 100
Conv01	7 x 7 x 96
Conv01	5 x 5 x 192
Conv01	3 x 3 x 512
Conv01	2 x 2 x 1024
Conv01	1024
Conv01	256
Conv01	K

a range of values between -1 and 1.

3.3.2 Loss Function

Let *C* be a set of *K* cluster pose centroids obtained via k-means clustering. Let **(D, P)** be a pair of depth image and a label pose, then the loss function for this regression task would look as follows:

$$Loss = (1 - \alpha) \cdot L_r(C \times g(D, \theta), p) + \alpha \cdot ||g(D, \theta)|| \tag{3.1}$$

Two parts of the loss function include a residual loss and an L_1 regularization. g refers to the neural network approximator, which takes in a depth image D, and has parameters. g predicts the weight matrix $\boldsymbol{w}$, and this matrix is multiplied with each column vector of *C*, in order to approximate the right 3D pose. $\boldsymbol{p}$ refers to the vectorized version of the ground truth pose **P**. The residual loss function used in this case can be defined by an L_1 smooth loss or a huber loss. Huber loss was also substituted by a basic mean-squared loss function, but sparsity introduced by the Huber loss led to better results, as also claimed by the authors in [37].

3.3.3 Inference

Given a test sample image, it is first pre-processed and resized to the input size of this network. A forward pass provides the weight matrix, and a final pose is estimated by multiplying this weight matrix to the linearly combine pose cluster centroids. This can be understood as a single view inference. For more than one cameras, results inferred from each depth image are again combined with appropriate weights in order to obtain even refined results for 3D estimation. A more detailed discussion on results obtained with this implementation is provided in the Results and Discussions section of Chapter 5.

CHAPTER 4

PROPOSED METHODOLOGY

A new methodology to solve 3D Pose Estimation in world coordinates is proposed, which takes in a point cloud as an input, and learns to obtain a set of 3D points which represent that person's skeleton. The idea for considering point cloud as model's input arises from the fact that it has the ability to provide a sparser representation of pixels present in a depth image. Therefore, a pointcloud would require much less space for any intermediate computation as compared to depth images. On the other hand, an established way to represent 3D data is voxelized grids, which essentially discretize a given pointcloud in a predefined set of values. Although voxels could be much sparser than pointclouds, but the process of discretizing pointclouds in the first place is cumbersome, and results in elongated wait times. Also, processing voxel grids in deep learning methods requires use of 3D convolutions, which again are very heavy in both required time as well as space in computations. Therefore, this work proposes a Point-Based Pose Estimator (referred as PBPE in future parts of the document), which efficiently process pointclouds from single and multiple views to estimate poses in 3D.

4.1 Point-Based Pose Estimator

4.1.1 Pointnet architecture

PointNet [44] is a pioneering work which understands a spatial encoding of points in 3D. This model takes in a fixed number of unordered point sets, and performs object recognition and semantic segmentation with remarkable accuracy and efficiency. PointNet merges local information about points with global information about the object's shape and appearance, within a single unified framework.

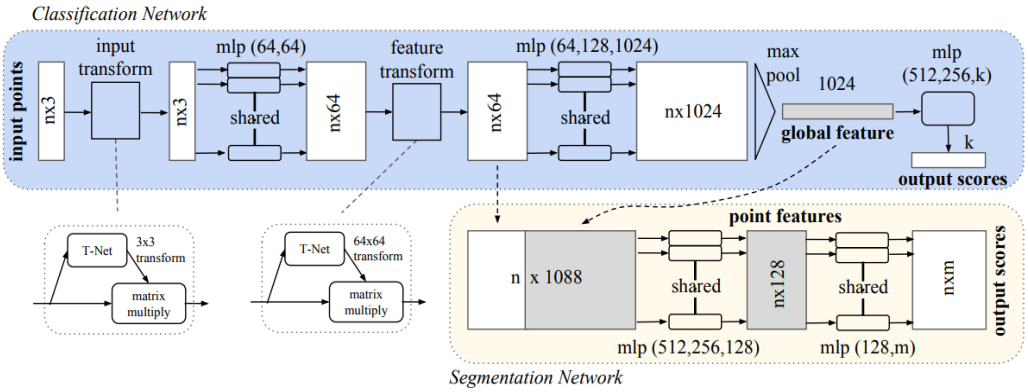


Figure 4.1: Architecture of PointNets for classification and segmentation

In the initial stages of design, each point of the group is processed individually to extract a powerful latent representation, and then, outputs from these layers are passed through a custom max-pooling layer, which performs aggregation to extract global attributes about that object. Input format of the point clouds is easy to apply any kind of rigid transformations. Therefore, PointNets also use a spatial transformer network to canonicalize the data before feature extraction. An overview of the PointNet pipeline as described in their paper can be viewed in Fig. 4.1.

The input layer takes in a point cloud with n points, which is declared at the time of training graph declaration. Once n is fixed, each of these n points are treated as a 2D input and the input vector resembles the following dimensions - $batchsize \times n \times 1 \times 3$. The initial (per-point) feature extraction is realized with the 2D convolutions, where 3 channels are actually the x, y, z coordinates of that specific point. This is also termed as a shared - multi-layer perceptron in the paper. The details of shared multilayered perceptrons are presented in a detailed visualization in Fig. 4.2.

Depending upon the application, pointnets use these local and global features to obtain context from each of these. For example, the global features are passed into a set of fully connected layers to obtain class of the object, and also, both local features concatenated with global features are passed into another set of shared multi-layer perceptrons to get pointwise labels. For each shared mlp layer realised with 2D convolution, batchnorm is

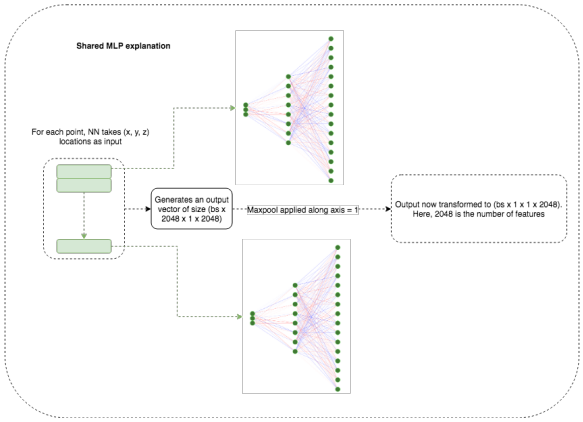


Figure 4.2: Explanation of the Shared MLP Layer

used followed by ReLU activations. For regularization, dropout layers are used in the last set of fully connected layers in the classification network.

The T-Nets in Fig. 4.1 before feature extraction are responsible for making input point-cloud canonical. Once learned, this T-net is actually performing a rigid transformation for the input pointcloud. Inspired by this, a next step of transformation is designed for feature sets too. An aligned feature space would have a lot more dimensions than that of the input point clouds, and this makes optimization a lot more difficult for the feature transformer network. In order to pacify this effect, a regularization is introduced in the softmax loss function, which basically enforces the basis for *feature vector space* to become an orthogonal matrix. This ensures that feature transformation is a good approximation of variations in the feature space.

Farthest point sampling is used in order to get most meaningful sample points for each input point cloud candidate. Once a uniform set of points is obtained, it is then normalized with min-max normalization, such that all points in the pointcloud fall between the values [-1, 1]. The minimum and maximum values are calculated by parsing all samples in the training dataset.

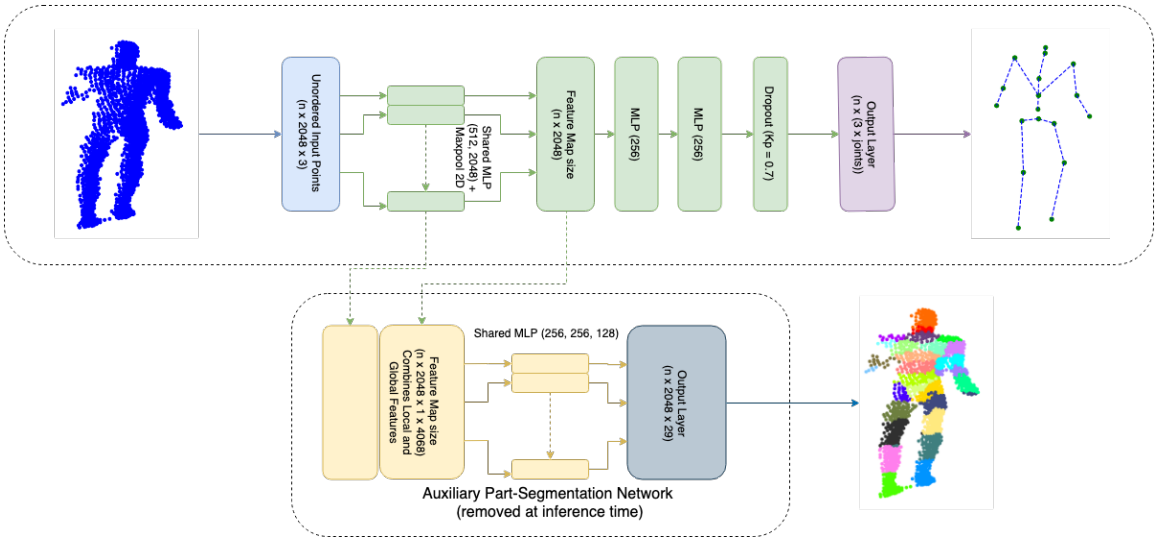


Figure 4.3: Details of layers in the proposed PBPE model.

4.1.2 Point Based Pose Estimation (**PBPE**)

Inspired by the PointNet architecture, a regression algorithm is proposed to estimate joint locations in the 3D world space. The main motivation behind developing a single stage pipeline with pointclouds is to make 3D pose estimation efficient in both space and time. Initial layers of the PBPE network follow the same pattern as PointNets. However, PBPE focuses on inferring pointclouds from the same camera viewpoint as a continuous feed. Therefore, initial T-net modules in the original PointNets are omitted in the PBPE design. This also helps in reducing overall number of learnable parameters. A detail of layers used in the regression net can be seen in Fig. 4.3.

Additional Semantic Segmentation Pipeline

Since the main objective of PBPE is estimating poses, the pipeline for semantic segmentation is not required inherently. Therefore, the first iteration of this design did not include semantic segmentation at all. Network looked like just the upper half of Fig. 4.3. Said design had problems converging directly, because of max-pooling aggregation in the intermediate layers made it lose almost all local context. For classification tasks, this seemed to

be the right thing to do, but for regression of joint locations, model did provide some kind of local region understanding. Deep Learning community often resorts to auxiliary losses in order to train models, in order to keep up the necessary gradient flow. Gradients for the *regression-only PBPE* subsided early in the training process. It would just give out one standard guess of an upright pose for all the pointclouds it inferred. Thus, there was a need for an intermediate regularization, which not only kept the gradients from dying, but also emphasized learning more about the human body locally. Therefore, an auxiliary segmentation pipeline was introduced, which in part was inspired from all the two-stage processes described in previous chapters. An important point to note here is that the segmentation module of the PBPE can be kept or removed at inference time based on individual applications. When only pose estimation is the priority, then the inference module would only consist of the truncated pose estimator, ensuring the primary objective of building PBPE, space and time efficiency. The automated annotation methodology described in previous chapter is directly extended to the point cloud case. It is even more convenient to use in this case because extra efforts in projecting an image to pointcloud back and forth are reduced. A detailed analysis of software, hardware and runtime requirements to train and test this network are provided in the next chapter of Results and Discussion.

CHAPTER 5
RESULTS AND DISCUSSION

This chapter describes all the experiments which were conducted throughout the course of this thesis. First section describes all the preprocessing stages involved in preparing datasets for different algorithms. This also provides details of how the real and synthetic data were fused in order to form one giant dataset. The second section dives deep into the results obtained in two-stage methods, and provides qualitative analysis of how the proposed annotation mechanism helps improve the existing state of the art results in two-stage pose estimation. Third section focuses on results obtained by implementation of the recent Deep Depth Pose algorithm, and finally, the PBPE section provides insights about how this single stage improves upon existing techniques in terms of efficiency, with reasonable trade-offs in pose estimation accuracy.

5.1 Dataset Splits and Pre-Processing

Berkeley MHAD has a total of 137684 depth images acquired by each of the two Kinect cameras used in their experiment, which are synchronized for consistency. The dataset does not provide any train-test split officially. Previous works have used a k-fold cross validation and custom train-test splitting to study this dataset for multiple experiments [34], Since this work proposes improvement in existing approaches, the two-stage approach is evaluated for two types of train-test splits. The first method selects 11 out of 12 subjects for training and one person is left out for testing. The pattern of leave-one-out validation is realized by selecting one person for testing at a time, in a total of 12 experiments. Results averaged out for all these experiments are reported as the final mean and median errors. Since leave-one-out approach is very slow to generate final results, another method used to split data is to randomly select 25 % of the MHAD dataset as our test set, keeping the rest for training. As

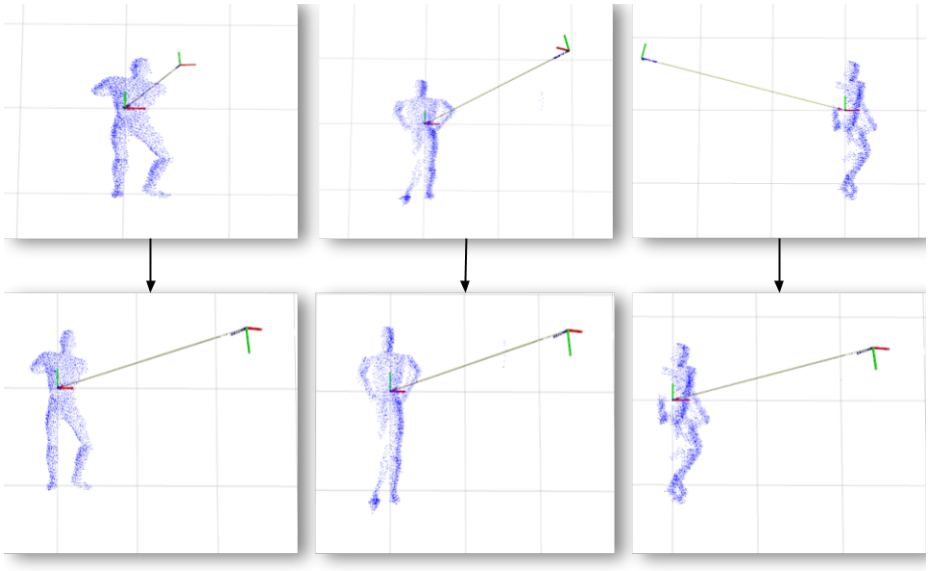


Figure 5.1: Fusing real and synthetic data: Each frame can have a different camera location in synthetic UBC. Thus, each single view pointcloud is transformed to camera coordinates of the real camera in MHAD.

described in more detail in Chapter 3, the synthetic data already comes with a distribution over train, test and validation splits, with a total of 177177 training images combined from 3 camera viewpoints, and 57057 images each for testing and validation.

5.1.1 Fusion of Real and Synthetic Data

In order to understand how synthetic data can be leveraged to improve performance of algorithms on the real test data, it was important to understand how real MHAD and synthetic UBC could be combined. First of all, relative position of subjects from the world coordinates in that frame is different as reported in individual datasets. Therefore, once these two datasets, were combined, statistically, the combined datasets would have more than one modes for all three dimensions. Since the end goal is to improve inference on real data, fusion resorts to transforming all the synthetic data into real camera coordinate system. The diagram in Fig. 5.1 shows the same.

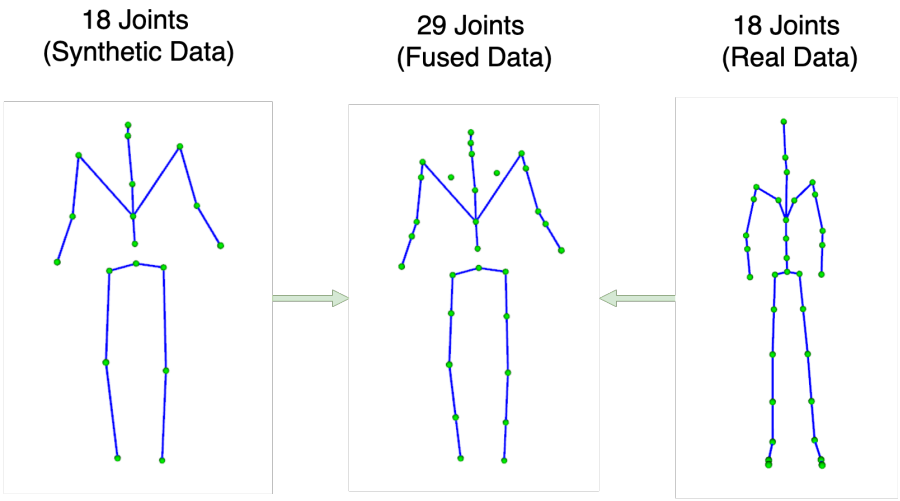


Figure 5.2: Fusing two datasets also requires that both abide by a universal definition of skeleton. The synthetic skeleton is interpolated, whereas the real skeleton is down-sampled.

Next challenge in the fusion of real and synthetic datasets was that real data was recorded with a ground truth of 35 different joint locations, whereas the synthetic data was generated only to provide 18 joint locations. More keypoints introduce more precision in practical used cases of keypoint estimation. Therefore, approximate additional joint locations were interpolated in the synthetic dataset. it can also be observed from Fig. 5.2 that some joints in the real MHAD Dataset are redundant, like the two pairs present at toe-tips, and one pair repeated at the fingertips. After all these practical considerations, a total of 29 joint locations were finalized as universal skeleton representation of real, as well as synthetic dataset.

5.2 Two-Stage Method

Table 5.1 reports the results obtained with automated annotation on the real and synthetic datasets. Under the column of Training datasets, numbers enclosed in the parentheses represent the number of body regions used to train first stage of the two-stage methods. First stage is semantic segmentation, which means a depth image is passed into the model and multiple pixel-wise labels for body region are inferred. During the train cycle, Adam optimizer is used to backpropogate the gradient values. Initial step size was set to be 10^{-3},

and then it was decayed after every 5000 steps, by an order of 10. Batch size was 10 for depth images of the dimensions 224 X 224. Parentheses in the second column are number of joints inferred to obtain the full skeletal model of the human body. A few important trends which can be observed from results in the table are as follows:

- ***Model Feasibility***: The authors of [34] do not provide their exact training model. As mentioned in Chapter 3, a slightly modified version of their model is implemented. In order to verify dependability of this modified version, the exact same data was used to train this model, as reported in [34]. The training data had 44 synthetically annotated body regions, and a total of 18 joints had to be inferred. Due to added skip connections and a more uniform upsampling and down-sampling criteria, this new model performs better, improving the pose estimation accuracy by around 17 %. Thus, this new implementation is kept as the base model for comparison in further experiments.

- ***Automatic Annotation***: Now, the effects of automatic annotation are studied. First, the automatic labels were generated for the synthetic data, and now with a total of 18 body regions, the mean-squared errors of pose estimation on the synthetic data increased from 4.7 cm to 5.5 cm. This can be related to the idea that more number of regions provide more structure to understanding the human body, and hence final pose estimation results get slightly worse with lesser intermediate regions.

- ***Number of Regions and Pose Estimation Accuracy on Real Data***: Next, the semantic segmentation model trained on synthetic UBC with 18 regions is used to infer these regions on the real MHAD data. With 18 inferred regions, performance decreases as compared to the original work in [34]. In order to concretize the importance of number of body regions, UBC data was interpolated as described above, in order to extract 29 regions now. An increase in number of classes for the first stage eventually improved pose estimation results. This proves the hypotheses about

Table 5.1: L2-Norm Errors of joint locations for Two-Stage Method.

Training Dataset	Testing Dataset	Annotation	Median Error-cm	Mean Error - cm
Our Experiments				
UBC-Hard(44)	UBC-Hard(18)	Synthetic	**2.95**	**4.70**
UBC-Hard(18)	UBC-Hard(18)	Automated	3.32	5.5
MHAD(35)	UBC-Hard(18)	Automated	6.68	11.04
UBC-Hard(18)	MHAD(35)	Automated	5.06	6.6
UBC-Hard(29)	MHAD(35)	Automated	4.38	5.36
MHAD(35)	MHAD(35)	Automated	**3.57**	**4.43**
Shafei et. al				
UBC-Hard(44)	UBC-Hard(18)	Synthetic	–	5.64
UBC-Hard(44)	MHAD(35)	Synthetic	–	5.01

number of regions fairly enough. Although the results are better than those with 18 intermediate regions, they still lag behind the existing best results (5.01 cm) by a small margin.

- ***Automated Annotations of Real Data***: Finally, the automatic annotation was applied on the real MHAD data directly. This representation of body regions substantially improved performance on the real data. Although increasing the number of body regions in synthetic data improved performance, but using synthetic data involved some amount of bias, which can be overcome with the help of automated labelling in the real data itself. The final mean squared error obtained on inferring on real testing data reduced to 4.43 cm, which is about 10% less than the present state of the art.

5.3 Single-Stage Methods

Along with two-stage methods, a version of the single stage method called the deep depth pose (DDP) [37] was also implemented for baseline results. This section first talks in detail about results obtained on trained models provided by the authors in [37] and then describes in detail about pros and cons of the proposed PBPE approach.

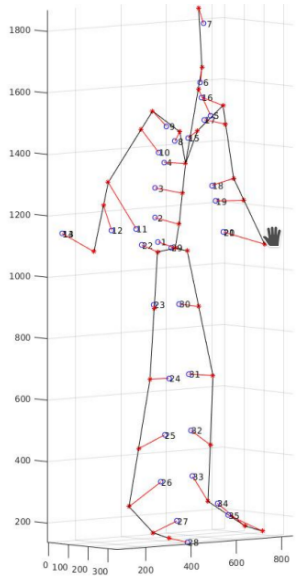


Figure 5.3: Errors reported are sums of L-2 norms between ground truth and predictions at each joint location.

5.3.1 Deep Depth Pose

Trained models for a different dataset (ITOP) [47], are provided by the authors of DDP. These were used to reproduce their results on the ITOP test dataset. Also, the authors do not provide a training code, but most of the training strategies are clearly stated in the paper [37]. All the network details were chosen strictly as mentioned, and implemented in the Keras framework. For the synthetic UBC dataset, K-means clustering was first done on the training set poses, and a total of 100 poses were obtained as the cluster centroids. These poses would serve as the bases for a vector space of poses. Observed results from these experiments are stated along side the claimed results in this paper in the Table 5.2. Due to a mismatch in these results, the authors were contacted, and they provided help by explaining the training procedure. The design and training strategies were reiterated, but the errors, did not come down to any reasonable figures.

Table 5.2: Errors claimed vs observed for DDP for the two Datasets

Dataset	Claimed Error (cm)	Observed Error (cm)
UBC-Hard	3.20	17.89
ITOP Dataset	2.93	15.50

Table 5.3: Performance Comparison of PBPE with other Single Stage Methods

Algorithm	Input Type	Input Dimensions	No. of Parameters	Inference Time
DDP	Depth Image	100 x 100 x 1	12.4 M	6000 fps
V2V Posenet	Voxel Grid	48 x 48 x 48	3.4 M	35 fps
PBPE (Ours)	Point Cloud	2048 x 3	1.8 M	192 fps

5.3.2 Point-Based Pose Estimation

This subsection first discusses about the exact computation time and space required by the proposed PBPE model. As can be observed in the Table 1, PBPE is the most efficient in terms of space requirements. Total number of learnable parameters in PBPE is 1.8 Million as opposed to V2V PoseNet [38], which operates on voxel grids of 3D Humans. The Deep Depth Pose [37], which operates directly on depth images has a total of 12.4 Million parameters, hence comparatively difficult to deploy in memory-constrained used cases. In terms of inference time, PBPE again infers much faster than V2V PoseNet, and outputs poses runs at a frame rate of 192 fps on an NVIDIA GTX 1080, which is much faster than real-time. The voxel-based model runs at a rate of 35 fps on an NVIDIA Titan X (which is already around 30% faster than a GTX 1080 [49]), but the results reported in their paper are achieved by an ensemble of 10 models, and therefore, both its space and time requirements are effectively 10 as much as reported. The DDP Model is the fastest in terms of runtime (6k fps), and these results are reported on an NVIDIA Titan XP GPU.

To gauge the performance of this technique, first of all this model was trained on synthetic data and real data individually. Then, after fusion of the two datasets, this model was trained with two different strategies in order to understand the end impact of this fusion on inferring real data. First, the model was trained purely on synthetic data, and then a fixed set amount of real data was used to fine-tune this model. In another set of experiments, the

Table 5.4: Pose Estimation errors on single view and multiple view pointclouds

Dataset	Single/Multiview	Mean Error (cm)	Median Error (cm)
UBC-Hard	Multi	5.59	4.28
MHAD	Multi	3.92	2.90
UBC-Hard	Single	7.59	5.12
MHAD	Single	7.46	3.31

Table 5.5: Results obtained for training with increasing additional synthetic data.

Dataset	Mean Error (cm)	Median Error (cm)
MHAD only	3.20	2.54
MHAD + (33 % Synthetic Data)	2.87	2.27
MHAD + (66 % Synthetic Data)	3.49	2.74
MHAD + (100 % Synthetic Data)	3.79	3.07

model was trained completely on real data first, and then some synthetic data was added, with the purpose of providing more generalization.

For all the experiments, learning rate was set to be 10^{-3} in the beginning of the training process, and then reduced with an exponential decay rate of 0.5 at each epoch. Also, the value of learning rate was clipped at 10^{-5}, so that the model does not get stuck in local minima. Results obtained in both these strategies are reported as follows:

Individual training: Single View and Multi-View

The PBPE was trained individually on the synthetic training set. First, all the three depth images were combined to form a single unified pointcloud. In the initial pre-processing, each pointcloud was subsampled to 2048 points in 3D. Then, a batch of 32 point clouds were passed into the network at each step. It can be observed from the Table 5.4 that results for multi-view pose estimation are much better than those of single view pose estimation. This inherently confirms the hypotheses that more data points in training set lead to better decisions. However, observing the results on single view is also important, as most real world setups would depend on single view pose estimation only, due to difficulties in real-time synchronization.

Mixed training: Single View

- ***Train with Real and Synthetic Data Together from scratch***: After individual experiments, focus of this work shifts towards how fusing real and synthetic data will be useful for improving the overall performance on real data, as explained in section 5.1.1. The new transformed dataset has a total of 29 joint locations now. First, the PBPE model is trained on real data only. It can be observed from the table 5.5 that without any additional synthetic data, and training extensively for a total of 30 epochs, the mean L2 Error in 3D Human Pose Estimation is 2.54 cm. Now, adding synthetic data demonstrates ultimately getting better results when inferring on the real MHAD dataset. As it has been clarified earlier that poses in synthetic data are more varied, adding synthetic data at train time provides augmentation with more variations, and thus helps the model generalize better than before. However, if the amount of added synthetic data keeps increasing steadily, then the accuracy in pose estimation on real data starts decreasing after a specific threshold. Hence, this pattern proves the theory of Bias-Variance Trade-off, where introducing more noise might also introduce more bias in the training procedure itself.

- ***Pre-Train on Synthetic, Finetune on Real Data***: Another set of experiments were performed to understand how abundant synthetic data can bring down necessity for the amount of real training data for applications. Therefore, the PBPE model is first trained on all synthetic data for a total of 10 epochs. The performance of these models are far from good on the real data, probably because of the slight approximations on joint locations made at the time of fusing these two datasets. However, this model has already learned to extract powerful features on the synthetic data itself. Now, using this pre-trained model, a very small amount of real MHAD data is obtained by random sampling from the training set, and the weights are fine-tuned for inference on the real data. It can be observed that with as less as 5% of the actual training

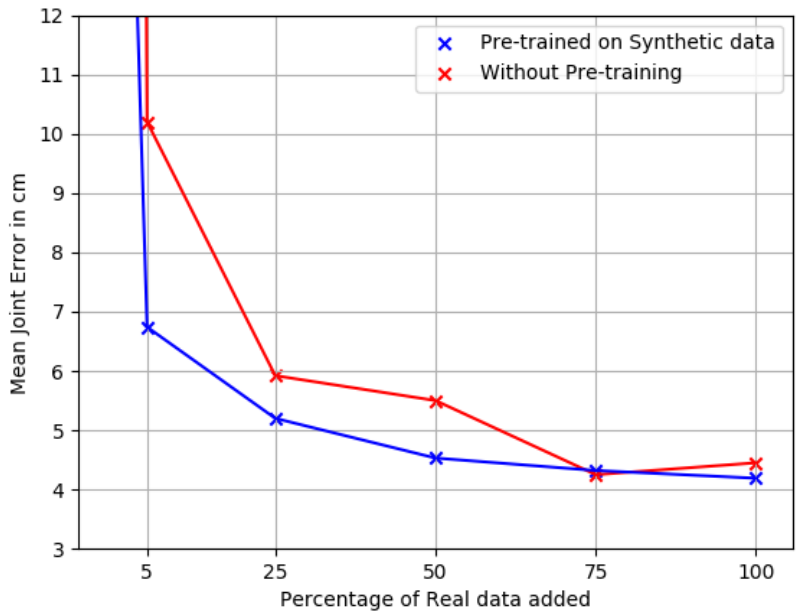
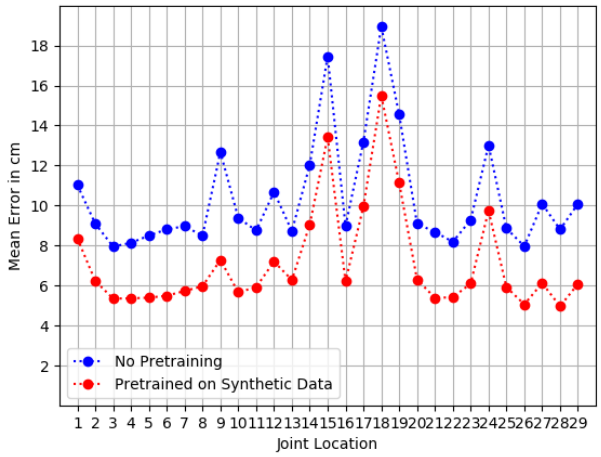
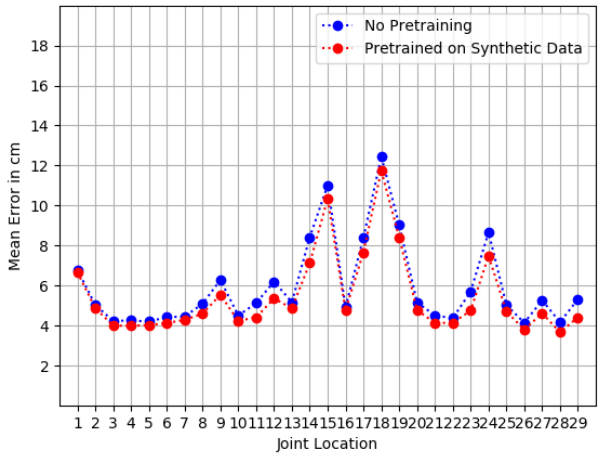


Figure 5.4: Error Comparison of pre-trained vs base model for increasing amount of train data.

dataset, the model starts giving error as low as 6.74 cm. On the other hand, if a new model is trained from scratch, with the same amount of real data, the error observed is more than 10 cm. A detailed visualization is also provided in the Figure 5.4, where all the results presented are obtained after 6 epochs. As a side note, better accuracies are possible if trained for more number of epochs, but right now the focus is on observing the trend that when this model is constrained to less than 50% of the available training data, pre-training has a huge impact on pose estimation accuracies. This analysis confirms the claim that with better approaches of generating realistic synthetic data, dependency on well-annotated real data would keep decreasing.



(a) Mean Errors for each joint when using only 5% of training data



(b) Mean Errors for each joint when using only 25% of training data

Figure 5.5: Overall errors and gaps within two sets decrease on increasing real train data

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

In this work, the impact that relevant synthetic data and semi-supervised annotation can have on learning about real world problems is studied through the lens of 3D Human Pose Estimation. A single-stage method for inferring 3D poses of humans, called Point-Based Pose Estimation (PBPE) is presented which outperforms existing techniques in space and time efficiencies, and is suitable for real-time applications. First, an automatic annotation mechanism for labeling limb regions is developed. This annotation on real data enables existing two-stage methods to learn directly on real data and improves their performance. Next, a new point cloud based pose estimation algorithm is developed, which also leverages automatic annotation performed in the first step. In the end, a real and a synthetic dataset are fused by modifying some properties like number of joints in the skeleton and a universal camera coordinate transformation. A detailed study confirms that incorporating more complex synthetic data in training, can be beneficial in multiple ways. One one hand, it provides a controlled mechanism for data augmentation, helping models generalize better with added noise. On the other hand, it enables powerful pre-training, reducing the cost and time required to annotate real data.

In future, this work can be extended in multiple directions. One, this algorithm can be adapted to learn poses of other rigid and articulated bodies like robotic manipulators, and hands of humans from 3D data. Second, actually deploying this algorithm on embedded devices and understanding the practical limitations and failure cases in a real-world scenario would be beneficial for a structured benchmarking against the existing Kinect algorithm. Third, a very interesting extension of this work would be to leverage studies in Domain Adaptation/Generalization to further explore how new real unlabeled 3D cloud data can be understood with available synthetic labels.

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 770–778.

[3] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. arXiv: 1703.06870.

[4] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. arXiv: 1506.02640.

[5] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2017, pp. 3645–3649.

[6] N. Wojke and A. Bewley, "Deep cosine metric learning for person re-identification," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2018, pp. 748–756.

[7] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3d traffic scene understanding from movable platforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 1012–1025, 2014.

[8] V. Ramanishka, Y. Chen, T. Misu, and K. Saenko, "Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning," *CoRR*, vol. abs/1811.02307, 2018. arXiv: 1811.02307.

[9] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, and S. Birchfield, *NDDS: NVIDIA deep learning dataset synthesizer*, https://github.com/NVIDIA/Dataset_Synthesizer, 2018.

[10] I. J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *CoRR*, vol. abs/1701.00160, 2017. arXiv: 1701.00160.

[11] J. ORourke and N. Badler, "Model-based image analysis of human motion using constraint propagation," *IEEE TPAMI*, vol. 2, 6 1980.

[12] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE TOC*, vol. C-22, 1 1973.

[13] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 55–79, 2005.

[14] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[15] D. Ramanan, "Learning to parse images of articulated bodies," in *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, 2006, pp. 1129–1136.

[16] Y. Yang and D. Ramanan, "Articulated human detection with flexible mixtures of parts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2878–2890, 2013.

[17] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '14, Washington, DC, USA: IEEE Computer Society, 2014, pp. 1653–1660, ISBN: 978-1-4799-5118-5.

[18] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," *CoRR*, vol. abs/1603.06937, 2016. arXiv: 1603.06937.

[19] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime multi-person 2D pose estimation using Part Affinity Fields," in *arXiv preprint arXiv:1812.08008*, 2018.

[20] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, 2017.

[21] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[22] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3d human pose estimation," *CoRR*, vol. abs/1705.03098, 2017. arXiv: 1705.03098.

[23] C. Chen and D. Ramanan, "3d human pose estimation = 2d pose estimation + matching," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5759–5767.

[24] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei, "Integral human pose regression," in *The European Conference on Computer Vision (ECCV)*, 2018.

[25] D. C. Luvizon, D. Picard, and H. Tabia, "2d/3d pose estimation and action recognition using multitask deep learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[26] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1325–1339, Jul. 2014.

[27] H. Fang, Y. Xu, W. Wang, X. Liu, and S. Zhu, "Learning knowledge-guided pose grammar machine for 3d human pose estimation," *CoRR*, vol. abs/1710.06513, 2017. arXiv: 1710.06513.

[28] W. Yang, W. Ouyang, X. Wang, J. Ren, H. Li, and X. Wang, "3d human pose estimation in the wild by adversarial learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[29] G. Rogez, P. Weinzaepfel, and C. Schmid, "Lcr-net++: Multi-person 2d and 3d pose detection in natural images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[30] B. Tekin, P. Márquez-Neila, M. Salzmann, and P. Fua, "Fusing 2d uncertainty and 3d cues for monocular body pose estimation," *CoRR*, vol. abs/1611.05708, 2016. arXiv: 1611.05708.

[31] H. Rhodin, M. Salzmann, and P. Fua, "Unsupervised geometry-aware representation for 3d human pose estimation," *CoRR*, vol. abs/1804.01110, 2018. arXiv: 1804.01110.

[32] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '11, Washington, DC, USA: IEEE Computer Society, 2011, pp. 1297–1304, ISBN: 978-1-4577-0394-2.

[33] A. Hernandez-Vela, N. Zlateva, A. Marinov, M. Reyes, P. Radeva, D. Dimov, and S. Escalera, "Graph cuts optimization for multi-limb human segmentation in depth maps," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 726–732.

[34] A. Shafaei and J. J. Little, "Real-time human motion capture with multiple depth cameras," in *Proceedings of the 13th Conference on Computer and Robot Vision*, Canadian Image Processing and Pattern Recognition Society (CIPPRS), 2016.

[35] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi, "Accurate, robust, and flexible real-time hand tracking," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '15, Seoul, Republic of Korea: ACM, 2015, pp. 3633–3642, ISBN: 978-1-4503-3145-6.

[36] M. Ye, R. Yang, and M. Pollefeys, "Accurate 3d pose estimation from a single depth image," in *2011 International Conference on Computer Vision*, 2011, pp. 731–738.

[37] M. Marin-Jimenez, F. Romero-Ramirez, R. Muñoz Salinas, and R. Medina-Carnincer, "3d pose estimation from depth maps using a deep combination of poses," *Journal of Visual Communication and Image Representation*, 2018, In press.

[38] G. Moon, J. Chang, and K. M. Lee, "V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[39] J. Nio-Castaeda, A. Fras-Velzquez, N. Bo Bo, M. Slembrouck, J. Guan, G. Debard, B. Vanrumste, T. Tuytelaars, and W. Philips, "Scalable semi-automatic annotation for multi-camera person tracking," *IEEE Transactions on Image Processing*, vol. 25, no. 5, pp. 2259–2274, 2016.

[40] M. Andriluka, J. R. R. Uijlings, and V. Ferrari, "Fluid annotation: A human-machine collaboration interface for full image annotation," *CoRR*, vol. abs/1806.07527, 2018. arXiv: 1806.07527.

[41] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," *CoRR*, vol. abs/1709.07857, 2017. arXiv: 1709.07857.

[42] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *CoRR*, vol. abs/1703.09312, 2017. arXiv: 1703.09312.

[43] F. Chu, R. Xu, and P. A. Vela, "Learning affordance segmentation for real-world robotic manipulation via synthetic images," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1140–1147, 2019.

[44] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CoRR*, vol. abs/1612.00593, 2016. arXiv: 1612.00593.

[45] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09, Montreal, Quebec, Canada: ACM, 2009, pp. 41–48, ISBN: 978-1-60558-516-1.

[46] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[47] A. Haque, B. Peng, Z. Luo, A. Alahi, S. Yeung, and L. Fei-Fei, "Towards viewpoint invariant 3d human pose estimation," in *European Conference on Computer Vision (ECCV)*, 2016.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105.

[49] M. Poppin., "The titan x vs. the gtx 1080," *Babeltech Reviews*, 2016.